Module variables

The HTTP Core module introduces a large set of variables that you can use within the value of directives. Be careful though, as only a handful of directives accept variables in the definition of their value. If you insert a variable in the value of a directive that does not accept variables, no error is reported; instead, the variable name appears as raw text.

There are three different kinds of variables that you will come across. The first set represents the values transmitted in the headers of the client request. The second set corresponds to the headers of the response sent to the client. Finally, the third set comprises variables that are completely generated by Nginx.

Request headers

Nginx lets you access client request headers under the form of variables that you will be able to employ later on in the configuration:

Variable	**Description**
`$http_host`	Value of the *Host* HTTP header, a string indicating the hostname that the client is trying to reach.
`$http_user_agent`	Value of the *User-Agent* HTTP header, a string indicating the web browser of the client.
`$http_referer`	Value of the *Referer* HTTP header, a string indicating the URL of the previous page from which the client comes.
`$http_via`	Value of the *Via* HTTP header, which informs us about the possible proxies used by the client.
`$http_x_forwarded_for`	Value of the *X-Forwarded-For* HTTP header, which shows the actual IP address of the client if the client is behind a proxy.
`$http_cookie`	Value of the *Cookie* HTTP header, which contains the cookie data sent by the client.
`$http_...`	Additional headers sent by the client can be retrieved using `$http_` followed by the header name in lowercase and with dashes (`-`) replaced by underscores (`_`).

Response headers

In a similar fashion, you are allowed to access the HTTP headers of the response that was sent to the client. These variables are not available at all times—they will only carry a value after the response is sent, for instance, at the time of writing messages in the logs.

Variable	**Description**
`$sent_http_content_type`	Value of the *Content-Type* HTTP header indicating the MIME type of the resource being transmitted.
`$sent_http_content_length`	Value of the *Content-Length* HTTP header informing the client of the response body length.
`$sent_http_location`	Value of the *Location* HTTP header, which indicates that the location of the desired resource is different from the one specified in the original request.
`$sent_http_last_modified`	Value of the *Last-Modified* HTTP header corresponding to the modification date of the requested resource.
`$sent_http_connection`	Value of the *Connection* HTTP header defining whether the connection will be kept alive or closed.
`$sent_http_keep_alive`	Value of the *Keep-Alive* HTTP header that defines the amount of time a connection will be kept alive.
`$sent_http_transfer_encoding`	Value of the *Transfer-Encoding* HTTP header giving information about the response body encoding method (such as compress, gzip).
`$sent_http_cache_control`	Value of the *Cache-Control* HTTP header, telling us whether the client browser should cache the resource or not.
`$sent_http_...`	Additional headers sent to the client can be retrieved using `$sent_http_` followed by the header name in lowercase and with dashes (`-`) replaced by underscores (`_`).

Nginx generated

Apart from the HTTP headers, Nginx provides a large number of variables concerning the request, the way it was and will be handled, as well as the settings in use with the current configuration.

Variable	**Description**
`$arg_XXX`	Allows you to access the query string (GET parameters), where `XXX` is the name of the parameter that you wish to utilize.
`$args`	All the arguments of the query string combined together.

Variable	**Description**
`$binary_remote_addr`	IP address of the client as binary data (4 bytes).
`$body_bytes_sent`	The number of bytes sent in the body of the response (does not include the response headers).
`$bytes_sent`	The number of bytes sent to the client.
`$connection`	Serial number identifying a connection.
`$connection_requests`	The number of requests already served by the current connection.
`$content_length`	Equates to the *Content-Length* HTTP header.
`$content_type`	Equates to the *Content-Type* HTTP header.
`$cookie_XXX`	Allows you to access cookie data, where `XXX` is the name of the parameter that you wish to utilize.
`$document_root`	Returns the value of the `root` directive for the current request.
`$document_uri`	Returns the current URI of the request. This may differ from the original request URI if internal redirects were performed. It is identical to the `$uri` variable.
`$host`	This variable equates to the *Host* HTTP header of the request. Nginx itself gives this variable a value for cases where the *Host* header is not provided in the original request.
`$hostname`	Returns the system hostname of the server computer
`$https`	Set to on for HTTPS connections, empty otherwise.
`$is_args`	If the `$args` variable is defined, `$is_args` equates to `?`. If `$args` is empty, `$is_args` is empty as well. You may use this variable for constructing a URI that comes with a query string option, such as `index.php$is_args$args`. If there is any query string argument in the request, `$is_args` is set to `?`, making this a valid URI.
`$limit_rate`	Returns the per-connection transfer rate limit as defined by the `limit_rate` directive. You are allowed to edit this variable by using `set` (directive from *The Rewrite module*): `set $limit_rate 128k;`
`$msec`	Returns the current time (in seconds + milliseconds).
`$nginx_version`	Returns the version of Nginx that you are running.
`$pid`	Returns the Nginx process identifier.
`$pipe`	If the current request is pipelined, this variable is set to `p`, otherwise the value is `"."`.

Variable	**Description**
`$proxy_protocol_addr`	If the `proxy_protocol` parameter is enabled on the `listen` directive, this variable will contain the client address.
`$query_string`	Identical to `$args`.
`$remote_addr`	Returns the IP address of the client.
`$remote_port`	Returns the port of the client socket.
`$remote_user`	Returns the client username if they use authentication.
`$realpath_root`	Returns the document root in the client request with symbolic links resolved into the actual path.
`$request_body`	Returns the body of the client request, or `-` if the body is empty.
`$request_body_file`	If the request body was saved (see the `client_body_in_file_only` directive), this variable indicates the path of the temporary file.
`$request_completion`	Returns OK if the request is completed, an empty string otherwise.
`$request_filename`	Returns the full filename served in the current request.
`$request_length`	Returns the total length of the client request.
`$request_method`	Indicates the HTTP method used in the request, such as GET or POST.
`$request_time`	Returns the amount of time elapsed since the first byte was read from the client (seconds + milliseconds value).
`$request_uri`	Corresponds to the original URI of the request, remains unmodified throughout the process (unlike `$document_uri/$uri`).
`$scheme`	Returns either `http` or `https` depending on the request.
`$server_addr`	Returns the IP address of the server. Beware while using this, as each use of the variable requires a system call, which could potentially affect the overall performance in the case of high-traffic setups.
`$server_name`	Indicates the value of the `server_name` directive that was used while processing the request.
`$server_port`	Indicates the port of the server socket that received the request data.
`$server_protocol`	Returns the protocol and version, usually `HTTP/1.0` or `HTTP/1.1`.
`$status`	Returns the response status code.

Variable	Description
`$tcpinfo_rtt`, `$tcpinfo_rttvar`, `$tcpinfo_snd_cwnd`, `$tcpinfo_rcv_space`	If your operating system supports the `TCP_INFO` socket option, these variables will be populated with information on the current client TCP connection.
`$time_iso8601`, `$time_local`	Provides the current time in ISO 8601 and local formats respectively for use with the `access_log` directive.
`$uri`	Identical to `$document_uri`.

The location block

We have established that Nginx offers you the possibility to fine-tune your configuration down to three levels—at the *protocol* level (`http` block), the server level (`server` block), and the requested URI level (`location` block). Let us now detail the latter.

Location modifier

Nginx allows you to define `location` blocks by specifying a pattern that will be matched against the requested document URI.

```
server {
    server_name website.com;
    location /admin/ {
    # The configuration you place here only applies to
    # http://website.com/admin/
    }
}
```

Instead of a simple folder name, you can indeed insert complex patterns. The syntax of the `location` block is:

```
location [=|~|~*|^~|@] pattern { ... }
```

The first optional argument is a symbol called **location modifier** that defines the way Nginx matches the specified pattern, and also defines the very nature of the pattern (simple string or regular expression). The following paragraphs detail the different modifiers and their behavior.